CS342 Project – 2

Ümit Yiğit Başaran

21704103

28.03.2020

For this experiment, minB, avgB, minA, and avgA values are changed. “avgA >> avgB”, “avgA == avgB” and “avgA << avgB” cases are tested. Also, for each algorithms, the outputs of the program (average waiting time for each thread, and average waiting time in across whole program execution) is given as Figures based on specific cases.

Experiments

N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 1000, avgA = 1500

FCFS

```
Avg. waiting time for thread 1: 44
Avg. waiting time for thread 2: 57
Avg. waiting time for thread 3: 91
Avg. waiting time for thread 4: 61
Avg. waiting time for thread 5: 67
Avg. waiting time for FCFS algorithm with 5 thread: 64
```

Figure 1 – FCFS (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 1000, avgA = 1500)

SJF

```
Avg. waiting time for thread 1: 133
Avg. waiting time for thread 2: 173
Avg. waiting time for thread 3: 194
Avg. waiting time for thread 4: 183
Avg. waiting time for thread 5: 233
Avg. waiting time for SJF algorithm with 5 thread: 183
```

Figure 2 – SJF (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 1000, avgA = 1500)

PRIO

```
Avg. waiting time for thread 1: 117
Avg. waiting time for thread 2: 166
Avg. waiting time for thread 3: 247
Avg. waiting time for thread 4: 202
Avg. waiting time for thread 5: 295
Avg. waiting time for PRIO algorithm with 5 thread: 205
```

Figure 3 – PRIO (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 1000, avgA = 1500)

VRUNTIME

```
Avg. waiting time for thread 1: 123
Avg. waiting time for thread 2: 345
Avg. waiting time for thread 3: 183
Avg. waiting time for thread 4: 214
Avg. waiting time for thread 5: 225
Avg. waiting time for VRUNTIME algorithm with 5 thread: 218
```

Figure 4 – VRUNTIME (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 1000, avgA = 1500)

N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 100, avgA = 200

FCFS

```
Avg. waiting time for thread 1: 12265
Avg. waiting time for thread 2: 11628
Avg. waiting time for thread 3: 11468
Avg. waiting time for thread 4: 10811
Avg. waiting time for thread 5: 12269
Avg. waiting time for FCFS algorithm with 5 thread: 11688
```

Figure 5 – FCFS (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 100, avgA =200)

SJF

```
Avg. waiting time for thread 1: 7422
Avg. waiting time for thread 2: 7729
Avg. waiting time for thread 3: 5303
Avg. waiting time for thread 4: 4231
Avg. waiting time for thread 5: 9395
Avg. waiting time for SJF algorithm with 5 thread: 6816
```

Figure 6 – SJF (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 100, avgA =200)

PRIO

```
Avg. waiting time for thread 1: 565
Avg. waiting time for thread 2: 5491
Avg. waiting time for thread 3: 12426
Avg. waiting time for thread 4: 18936
Avg. waiting time for thread 5: 25486
Avg. waiting time for PRIO algorithm with 5 thread: 12580
```

Figure 7 – PRIO (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 100, avgA =200)

VRUNTIME

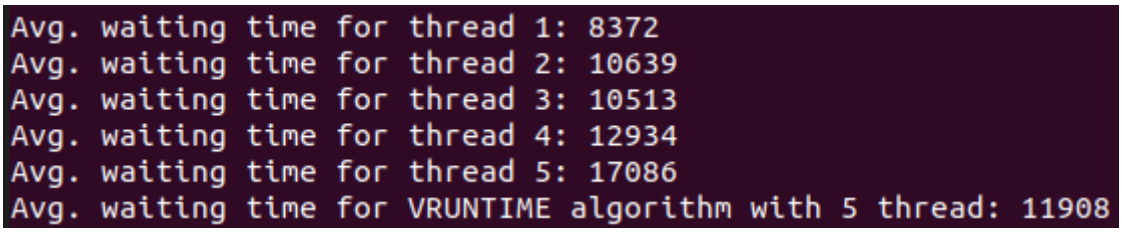

Figure 8 – VRUNTIME (N = 5, Bcount = 20, minB = 100, avgB = 200, minA = 100, avgA =200)

N = 5, Bcount = 20, minB = 1000, avgB =1500, minA =100, avgA = 200

FCFS

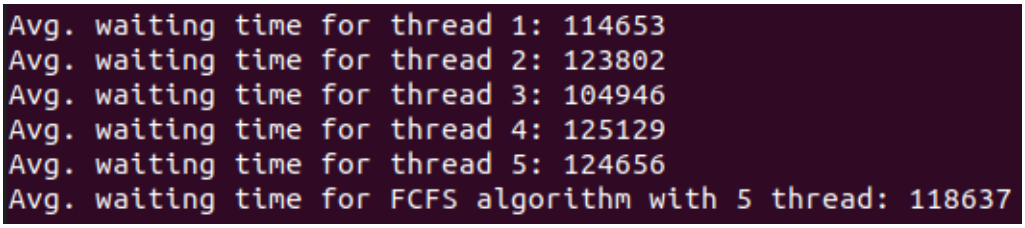

Figure 9 – FCFS (N = 5, Bcount = 20, minB = 1000, avgB = 1500, minA = 100, avgA =200)

SJF

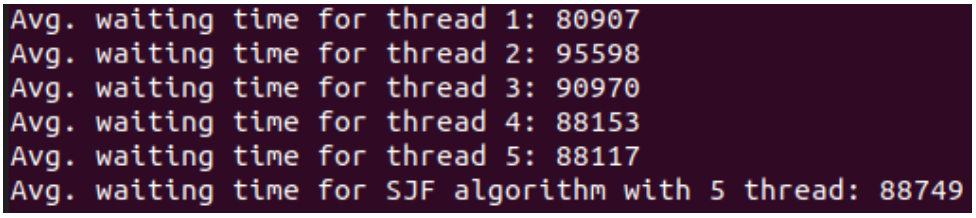

Figure 10 – SJF (N = 5, Bcount = 20, minB = 1000, avgB = 1500, minA = 100, avgA =200)

PRIO

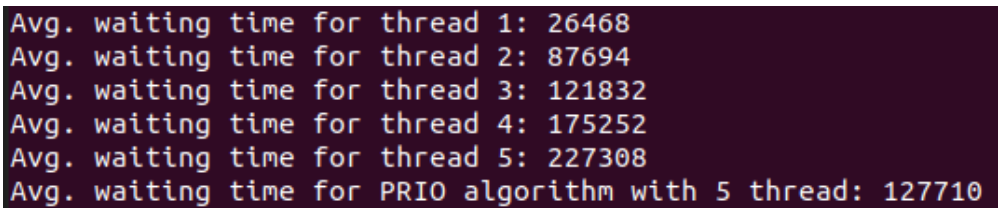

Figure 11 – PRIO (N = 5, Bcount = 20, minB = 1000, avgB = 1500, minA = 100, avgA =200)

VRUNTIME

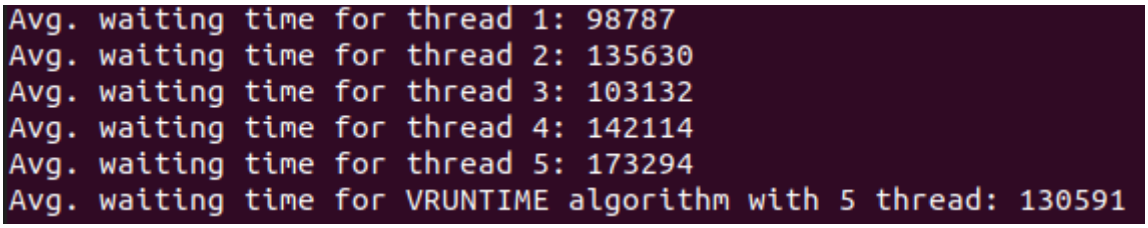

Figure 12 – VRUNTIME (N = 5, Bcount = 20, minB = 1000, avgB = 1500, minA = 100, avgA =200)

Conclusion

Algorithms → Parameters	**FCFS**	**SJF**	**PRIO**	**VRUNTIME**
minB = 100, avgB = 200, minA = 1000, avgA = 1500	**64 ms**	**183 ms**	**205 ms**	**218 ms**
minB = 100, avgB = 200, minA = 100, avgA = 200	**11688 ms**	**6816 ms**	**12580 ms**	**11908 ms**
minB = 1000, avgB = 1500, minA = 100, avgA =200	**118637 ms**	**88749 ms**	**127710 ms**	**130591 ms**

Figure 13 – Parameters and Algorithms Table

In conclusion, in this experiment, different **minB, avgB, minA, and avgA** values are tested for each algorithm. For each experiment, 100 burst nodes are created and scheduled based on selected algorithm.

When parameters are **100, 200, 1000, 1500**, respectively, the FCFS algorithm has the best waiting time value. Because the time between two concurrent bursts (**minA, avgA**) is greater than the time between each scheduling (**minB, avgB**) adjusted by the scheduler thread, the scheduler waits for a burst node to come to the run queue. So that, when a worker thread creates a burst node based on minB and avgB, the scheduler thread takes it from the run queue within a very short time. Therefore, the average waiting times for each thread and for each algorithm is very low as it can be seen from Figure – 1, 2, 3, and 4. Also, when the general average for each algorithm are compared among themselves, it can be seen that FCFS algorithm is the most optimum algorithm when avgA >> than avgB value (Figure - 13).

When parameters are **100, 200, 100, 200**, respectively, the SJF algorithm has the best waiting time value. When avgA and avgB is very close to each other. The waiting time values are highly increased for all threads and algorithms. Because scheduler thread is not faster than the worker threads, worker threads add lots of burst nodes to the run queue and while in this insertion operation because the linked list is synchronized, the scheduler threads wait to the other threads. At the end, the scheduler thread, select and simulate the execution of the burst nodes. Therefore, the waiting times higher than the previous case's waiting times. For that case the best result is taken from the SJF algorithm because the selection operation is done after insertion operations.

When parameters are **1000, 1500, 100, 200**, respectively, again the SJF algorithm has the best waiting time value. When avgB >> avgA, the worker threads create nodes a lot faster than retrieving nodes by the scheduler thread. So that, the waiting time of the nodes increases a lot. Also, because the time between creating nodes is much lower than selecting threads, the SJF algorithm gives the best waiting value.

For the PRIO and VRUNTIME algorithms, it can be seen that, the waiting time for the first thread (superior) is less then the last thread. Because the priority values are based on their thread ids. The VRUNTIME algorithm gives more close waiting times for each thread, while the PRIO algorithm doesn't give close waiting times especially when avgB >= avgA where it can be seen from the Figures in the previous pages.